УДК 004.7.056.53

АЛГОРИТМ И МЕТОДИКА КОНТРОЛЯ И УПРАВЛЕНИЯ АВТОРИЗАЦИЕЙ ПОЛЬЗОВАТЕЛЕЙ В WEB-ПРИЛОЖЕНИИ

А.В. ВЛАСЕНКО, П.И. ДЗЬОБАН

Кубанский государственный технологический университет
350072, Российская Федерация, г.Краснодар, ул.Московская, 2
электронная почта: Vlasenko@kubstu.ru, bulletproofpaul@gmail.com

Разработка алгоритмов, инструментов и методов авторизации пользователей в web – приложениях с использованием хеш-функций необходимы как для активной, так и для пассивной защиты данных идентификации и аутентификации пользователей разрабатываемого web-приложения. Даже в случае утери авторизационных данных пользователем, злоумышленнику не хватит временного ресурса на конвертацию их в исходный вид для дальнейшего использования, так как разработчик может регулировать время устаревания Cookie.
Ключевые слова: Web – приложения, идентификация пользователя, аутентификация, авторизация, коллизия, база данных, хеширование, криптостойкость.

Большинство из актуальных классов атак на web – приложения приходятся на этап авторизации пользователя, а именно процесс передачи идентификационных и аутентификационных данных от пользователя к БД web – приложения, минуя брандмауэры web – приложения и различные системы защиты, как программного, так и аппаратного уровней.

Одним из действенных приемов злоумышленника является получение данных авторизации из Cookie, даже если данные будут в захэшированном виде – на этом этапе смысл любой защиты пропадает. Если работать только с сессиями (при закрытии браузера все идентификационные данные обнуляются), то каждый раз при входе на часто посещаемый сайт или web – приложение придется вводить данные авторизации. Если учесть что у большинства пользователей несколько почт, разных паролей и логинов – то такие меры по защите приведут к элементарной записи всех паролей и логинов пользователей на материальный носитель – что менее всего защищается и смысл в защите вообще исчерпывается[1]. Есть смысл работать и с Cookie и с сессиями, грамотно распределив нагрузку. Представим форму, с помощью которой пользователь сможет передать нам свой логин и пароль на рисунке 1.

После ввода пользователем авторизационных данных, необходимо проверить его логин и пароль. Очевидно, данные авторизации не хранятся в чистом виде. Существуют различные, как криптографические, так и некриптографические хеш-функции. Представим требования к криптографическим хеш-функциям, отличающихся следующими условиями от остальных хеш-функций:

- стойкость к коллизиям 1-го рода: для какого-либо сообщения P должно быть невозможно в реальном времени подобрать другое какое-либо сообщение Q, для которого хеш-функция F(P)=F(Q);
- стойкость к коллизиям 2-го рода: должно быть невозможно, в реальном времени, подобрать такую пару сообщений (P, P'), хеш для которых одинаков;
- необратимость: для установленного значения хеш-функции A должно быть невозможно в реальном времени найти блок данных X, хеш-функция для которого F(X)=A[3].

Для проверки данных авторизации необходимо сравнить хеш введённого пароля с тем, что хранится в БД приложения, как представлено на рисунке 2.

Процесс хеширования пароля может проводиться с использованием алгоритма, которому отдаст предпочтение разработчик. Не стоит вникать в криптографическую реализацию всех хеш-функций, стоит обратить внимание на функции, к которым не найдены и не скомпрометированы коллизии. Например, алгоритм MD5 не актуален с 2013 года, а алгоритм bcrypt/scrypt остается актуальным. Если обратиться к российскому опыту, то для большинства проектов смена хеша на scrypt оказывается безболезненной.Так же, далее будет рассмотрено взаимодействие scrypt с криптографическим алгоритмом хеширования whirpool в алгоритме авторизации привилегированных пользователей.

К примеру, многие банки, для дистанционного банковского обслуживания, всё чаще выбирают решения, основанные на отдельном устройстве, которое генерирует хеши уже с заданной скоростью[2]. Значения

скорости перебора хешей (в мегахешах в секунду) которые были получены на современной видеокарте AMD Radeon 7870 представлены в таблице 1.

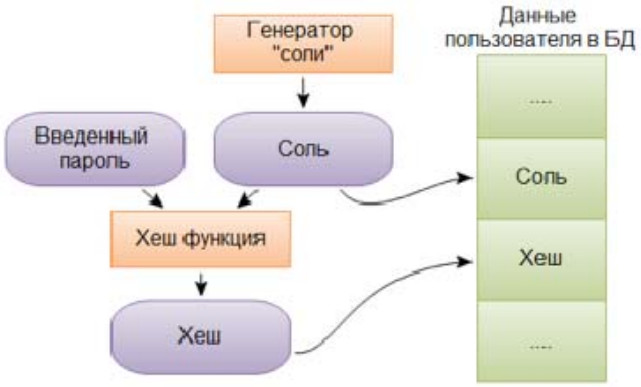

Рисунок 1 – Простой алгоритм процесса авторизации пользователя

Данный алгоритм являет собой реализацию методики контроля и управления авторизацией обычного пользователя, отдано предпочтение алгоритму с многочисленными коллизиями md5.

Таблица 1 – Значения скорости перебора хешей на современной видеокарте AMD Radeon 7870

Алгоритм хеширования	Скорость перебора хеша (М/с)
MD5	13200
SHA-1	4800
SHA256	1580
SHA512	170
NTLM	26300
bcrypt	0,0075

В авторской методике предлагается разделить пользователей, исходя из прав и полномочий пользования контентом web-ресурса, на обычных и привилегированных. Очевидно, что обычных пользователей значительно больше, чем пользователей с root-правами.

Исходя из рассмотренных методов авторизации пользователей, посредством многофакторной аутентификации с одной стороны, и

проанализированных уязвимостей и статистики атак с другой, были разработаны модель угроз и модель нарушителя.

В разработанном алгоритме генерируема и использована соль исходя из количества символов в логине пользователя. Для каждого пользовательского пароля (при его регистрации), будет генерироваться своя соль и записывать в базу рядом с паролем, чтобы использовать при хешировании и последующем сравнении хеша авторизационных данных пользователей.

Последовательность тестирования предложенного алгоритма:

- регистрация пользователя: md5 с паролем password и солью 8f*;
- получение его хеша, используя md5(md5('password') . '8f*');
- запись пользователя в таблицу, в результате:

1 | md5 | 84cd3e7ff13bbaed1c1db91671844bcc | 8f*

Хеш-функция, обладающая в рамках web-приложения популярностью у рядовых пользователей с одной стороны и имеющая огромное количество коллизий с другой, наоборот, достаточно часто дает одинаковые хэши для множества разных паролей. Так же, посредством уменьшения разрядности хеша до 32 бит можно добиться большего количества коллизий. К примеру, используются пароли длиной от 1 до 10 символов, содержащие цифры, латинские буквы разных регистров.

Таким образом, получается следующее количество значений паролей:

$\Sigma_{i=1..10}(59^i) = 519\ 929\ 111\ 116\ 169\ 700$,

т.е. порядка полуквинтиллиона -что даже меньше количества значений 64-битного хеша в 36 раз.

Из полученных данных, можно подсчитать примерное желаемое количество паролей на один 32-битный хеш, в результате получив порядка 120 миллионов паролей на хеш.

Этот метод защищает входные данные пользователей от радужных таблиц, потому что теперь каждый пароль имеет свою уникальную соль.

Злоумышленник должен создать 10 миллионов отдельных радужных таблиц, что было бы совершенно нецелесообразно.

Разработанная методика подразумевает шифрование закрытого контента тем ключом, а точнее – паролем, который точно не содержится в базе данных ни в открытом, ни в закрытом виде. От данного пароля в базе данных имеется только хеш пароля, полученный при помощи алгоритма с многочисленными коллизиями md5, которому соответствует еще несколько миллионов паролей и, соответственно, для каждого из этих паролей результат расшифровки закрытых данных будет абсолютно неудачным.

Плюсы разработанного алгоритма и метода авторизации рядовых пользователей:

- снижены требования к паролю пользователей;
- исключен метод подбора и дешифрования пароля пользователя;
- отсутствие возможности авторизации через сторонние сервисы;
- частный подход к алгоритмам шифрования.

Стоит отметить, что в случае смены пароля, пользователю предстоит очень трудоемкий процесс по перешифрованию закрытого контента, что уменьшает показатель удобства использования web- приложения, однако для данной группы пользователей данная перспектива не должна выглядеть пугающей, учитывая уровень ответственности.

Хеширование авторизационных данных пользователей в браузере, для передачи данных в зашифрованном виде и повторное на сервере – это один из ключевых моментов алгоритма, который представлен на рисунке 2.

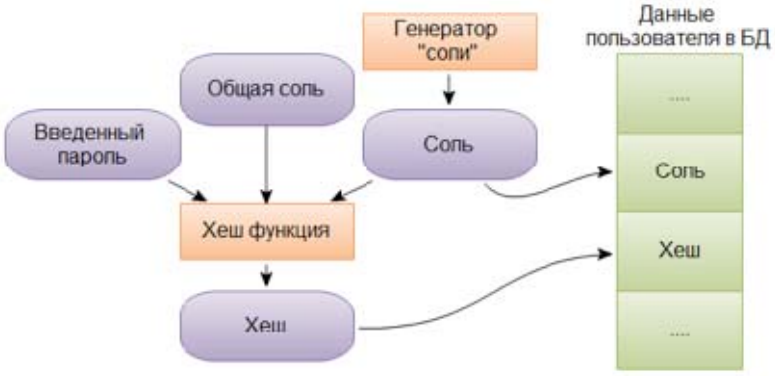

Рисунок 2 - Алгоритм сравнения введенных авторизационных данных пользователя с вычисленными данными на web-сервере

Поскольку в предлагаемом методе хешируется и используется соль и на сервере, то можно использовать имя пользователя (или адрес электронной почты), конкатенированный со специфичной для web-приложения строкой – id устройства пользователя, в качестве соли на клиентской стороне. Данный алгоритм дополнен разработанным инструментом – замедление хеш-функции. Цель – сделать хеш-функцию достаточно медленной, чтобы помешать атакам, но при этом довольно быстрой, чтобы не создавать видимой задержки при авторизации для пользователя.

Растяжение ключа реализуется при помощи специального типа хеш-функции, нагружающей ЦПУ. Простое итеративное хеширование хеш-кода пароля не достаточно, так как оно может быть распараллелено на аппаратным уровне и выполнено так же быстро, как и обычный хеш-код. В данном случае, используется стандартный алгоритм, такой как PBKDF2 или bcrypt.

Эти алгоритмы принимают в качестве аргумента коэффициент надежности или счетчик итераций. Эта величина определяет, насколько медленной будет хеш-функция.

Для настольного программного обеспечения или приложений для смартфонов самый лучший способ выбрать следующий параметр – это запустить короткий тест на устройстве, чтобы найти величину, при которой хеш-код вычисляется примерно полсекунды. Таким образом, авторское web-приложение будет максимально надежным, не влияя на работу пользователя.

На рисунке 3 представлена проверка правильности пароля при авторизации пользователя в рамках разработанной методики.

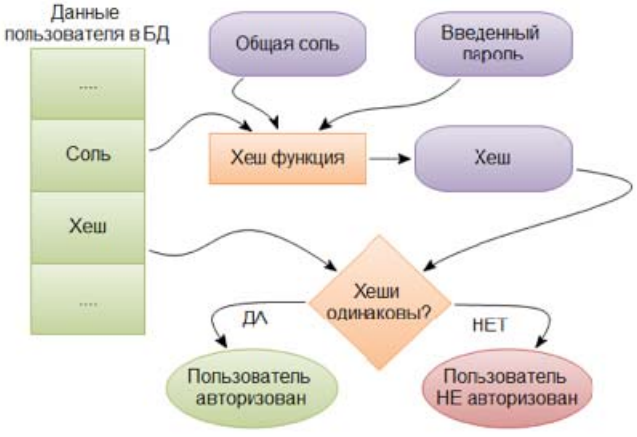

Рисунок 3 -Алгоритм проверки правильности пароля при авторизации пользователя web-приложения

Так как применяется хеширование растяжения ключа в web-приложении, то потребуются дополнительные вычислительные ресурсы, чтобы обрабатывать огромный объем запросов аутентификации, и что растяжение ключа облегчает выполнение атак типа «отказ в обслуживании» (DoS) на web-ресурс. Поэтому, в разработанной методике рекомендуется использовать растяжение ключа, но с небольшим значением счетчика итераций. Необходимо вычислить счетчик итераций, основываясь на наших вычислительных ресурсах и ожидаемой максимальной скорости запросов на проверку подлинности.

ЛИТЕРАТУРА

1. Кузнецов С. Д. Базы данных. Модели и языки. М.: Бином-Пресс, 2008.- 720с.

2. Веденьев Л.Т., Афанасьев А.А., Афанасьев А. Н. Аутентификация. Теория и практика обеспечения безопасного доступа к информационным ресурсам. Учебное пособие для вузов. Гриф УМО МО РФ. 2-е изд., 2012 – 552с.

3. Колисниченко Д.Н. PHP и MySQL. Разработка веб-приложений. 5-е изд., 2015 – 593с.

REFERENCES

1. Kuznetsov S. D. Bazy dannykh. Modeli i yazyki. M.: Binom-Press, 2008.- 720s.

2. Vedenev L.T., Afanasev A.A., Afanasev A. N. Autentifikatsiya. Teoriya i praktika obespecheniya bezopasnogo dostupa k informatsionnym resursam. Uchebnoe posobie dlya vuzov. Grif UMO MO RF. 2-e izd., 2012 – 552s.

3. Kolisnichenko D.N. PHP i MySQL. Razrabotka veb-prilozheniy. 5-e izd., 2015 – 593s.

ALGORITHM AND METHOD OF CONTROL AND MANAGEMENT AUTHORIZATION LOGIN THE WEB-APPLICATIONS

A.V. VLASENKO, P.I. DZOBAN

Kuban State Technological University,
2, Moskovskaya st., Krasnodar, Russian Federation, 350072,
e-mail: Vlasenko@kubstu.ru, bulletproofpaul@gmail.com

Development of algorithms, tools, and methods for user authentication in the web - applications using hash functions necessary for active and passive protection for data identifying and authenticating users of the developed Web-based applications. Even in the case of loss of authorization from the user, the attacker would not be enough time resource to convert them into original form for future use as a developer can adjust the aging time Cookie.

Key words: Web - application, user identification, authentication, authorization, collision, database, hashing, cryptographic.